SocioPath: In Whom You Trust?

Nagham Alhadad, Philippe Lamarre, Yann Busnel, Patricia Serrano-Alvarado, Marco Biazzini

SocioPath: In Whom You Trust?

Nagham Alhadad Philippe Lamarre Yann Busnel
Patricia Serrano-Alvarado Marco Biazzini

{Name.LastName}@univ-nantes.fr
LINA / Université de Nantes
2, rue de la Houssinière
BP 92208 – 44322 Nantes, France

Abstract

Distributed systems are getting more and more numerous, complex and used in a wide variety of applications. New solutions and new architectures arise (*e.g.*, clouds) that support new functionalities (*e.g.*, social networks). They pile up several software layers and, given that any software is directly dependent of the underlying layers, it can be unable to provide promised services whether any of these layers misbehaves. This evolution implies new non negligible *dependences* increasing with the number of actors involved in the system (*e.g.*, providers and users). Some dependences could be hidden by this layer stacking, implying a reduced transparency for users and a misunderstanding of her actual autonomy. We argue that users should be aware of the potential risks resulting from these dependences. To be able to deduce them, one should know the way the system works (architecture, involved resources, providers, participants, *etc*.). This would help to deduce the potential trust a user could or should have toward the system. We consider this of utmost importance, as professional efficiency and personal privacy could be compromised if untrusted actors control the access to others' resources. This work proposes SOCIOPATH, a generic meta-model that allows to expose hidden or implied relationships among participants in the digital world, which also introduce dependences at the social level. The notions presented in this approach are basics of many fields, like security, privacy, trust, sociology, economy and so forth. SOCIOPATH can be used in the evaluation process of a system as well as in its upstream design.

1 Problem definition

A large number of distributed systems arise nowadays that are more and more complex and used for a tremendous variety of applications. Actually, solutions proposed to users evolve toward new functionalities (*e.g.*, social networks), new architectures that support them (*e.g.*, clouds) and pile up several software layers. This evolution implies new non negligible *dependences* among providers and actors in the system.

When users need to choose a system, they are overwhelmed by the plethora of free and lucrative available options. To make a choice, they *evaluate* systems according to its perceived capability to satisfy their needs and the time they have to make their choice. Traditionally, the evaluation covers functional, technical and economical aspects. From the functional point of view, users analyze the quality of provided services and the user-friendliness of a system. Technical aspects orient the evaluation to operational criteria (*e.g.*, response time, reliability, availability, safety, security, *etc*.) but also to deployment and maintenance requirements. Moreover, economical aspects are considered, like the necessary investments to start up the system and to manage its long-term support.

From one user to another, those evaluation criteria have different weights and consequences and for the same applicative needs, different systems may be chosen. With a more technical approach, one may also consider that functionalities and performance of any software directly depend on underlying layers. If one of these layers misbehaves, the given software may be unable to provide the promised services. Thus, the whole system architecture characteristics should be taken into account while choosing the single component.

In general, users assume software developers and device manufacturers are competent and have the best intentions. Using a system entails the drawing of relationships of *trust* between users and providers and users are not always aware of those implicit relationships. A kind of "trust among participants" is *de facto* constructed, based

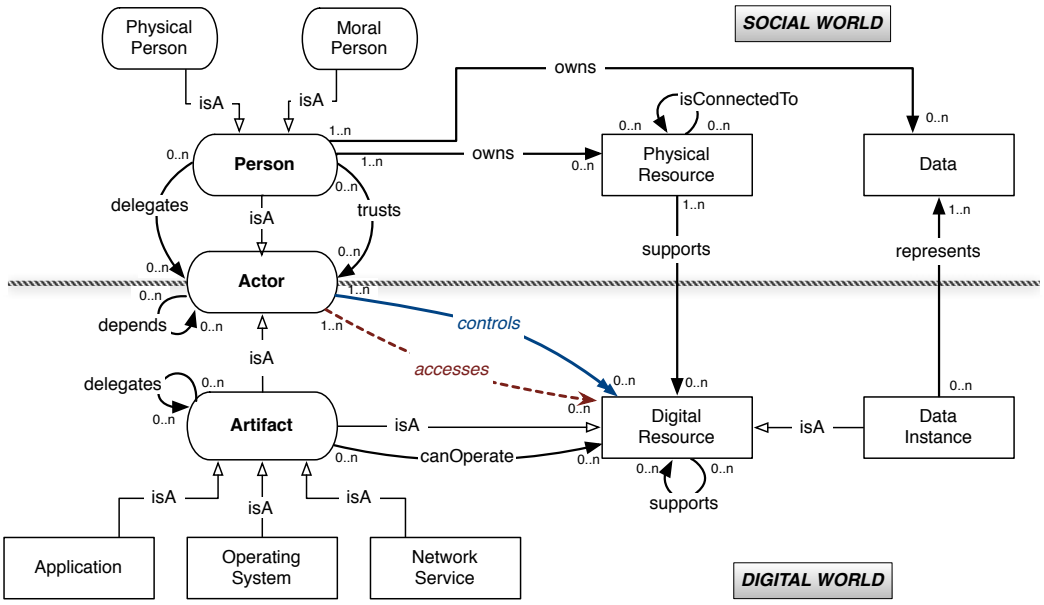

Figure 1: Graphical view of SOCIOPATH.

on a mix of more or less conscious factors such as the quality of their exchanges [1, 2, 3, 4, 5] or the trust toward resources (data, programs, communications, *etc.*) and providers [6, 7, 8, 9]. In this work, we consider that this necessary user's trust must be explicit. We argue that users should be aware of the potential risks resulting from their dependences on the system, either by means of public information or by their own deductions. To this end, the way the system works (underlying architecture, involved resources, providers, participants, *etc.*) must be explicit. This would help users to deduce the potential trust they should have toward the system.

We propose in this paper the meta-model SOCIOPATH. It allows to identify those relations among hardware and software components of a system that entail, in the social world, dependences among the actors that are involved in their availability, proper operating or use. The idea is that deduced relationships underline the potential repercussions of the trust users have toward the system in terms of security, privacy, social relationships, economy, *etc.* Thus, when assessing the suitability of a system, we should take into account functional, technical, economical but also dependence-related aspects. An extended paper given more details about SOCIOPATH and the state of the art is available in [10].

2 SOCIOPATH meta-model

SOCIOPATH may be seen as a tool that helps representing the reality of two worlds, which come together and interact between each other: the *social world* and the *digital world*. SOCIOPATH aims at providing a formalism to help the user to be aware of her relations in the social world (dependences on persons), as they emerge by the relations existing in the digital world (dependences on resources).

Figure 1 shows the graphical representation of SOCIOPATH. In the next we give a description of the nodes in the two worlds and the relations between them respectively in Sections 2.1, 2.2, 2.3. Some of the deduction rules and the important defintions are introduced in Section 2.4. Due to space constraints, only a subset of them is presented. All together are used to point out social dependences implied by the relations in the digital world.

2.1 The social world

The social world describes physical and moral persons (enterprises, companies, *etc.*), physical resources, data and the relations among them.

- *Person*: either a Physical or a Moral Person;
- *Data*: an abstract notion that does not necessarily imply a physical instance (*e.g.*, address, software, *etc.*);
- *Physical Resource*: hardware device (*e.g.*, PC, USB device, *etc.*).

2.2 The digital world

The digital world has nodes characterizing digital resources, artifacts, data instances, operating systems, networks services and applications.

- *Data Instance*: a digital representation of Data. It may be semantically equivalent to data that exist in the social world. For instance, a person has an address (Data) in the social world. Whenever she writes it in a file, she creates a semantically equivalent digital instance of her address in the digital world (Data Instance);
- *Artifact*: it may be an Application, an Operating System or a Network Service. We mean all of them to be "running software", thus considering them only in that they are being executed. By *Application*, we mean a whole running entity. It may be a single process or a group of processes that may be distributed in different locations, yet defining a single logically coherent entity;
- *Digital Resource*: an Artifact or a Data Instance;
- *Actor*: a Person in the social world or an Artifact in the digital world.

2.3 The relations in SOCIOPATH

Several relations are drawn in SOCIOPATH. We briefly describe them as follows.

- *owns*: it means ownership. This relation exists only in the social world;
- *isConnectedTo*: it means that two nodes are physically connected. This relation exists only in the social world and it is intrinsically symmetrical;
- *trusts*: relations of trust exist among persons and can be drawn from persons to artifacts. One of the future goals of our work is to assess if and how a given architecture "deserves" the users trust toward the system;
- *delegates*: a Person can delegate another Actor to perform some kind of access or control on a resource. The same concept of delegation can be implemented among artifacts (*e.g.*, in large databases distributed transactions are often performed by means of chains of delegations);
- *canOperate*: it means that the artifact is able to process, communicate, interact with the target resource. This ability may be given as a part of the artifact specification or deduced by some contingent property of the system (*e.g.*, an operating system only canOperate those files that are stored in a mounted partition);
- *accesses*: an Actor can access a Digital Resource (*e.g.*, the operating system accesses the applications installed on it, or a person who owns a PC that supports an operating system accesses this operating system). The access relations we consider are: read, write, execute;
- *controls*: an Actor can control a Digital Resource. There may be different kinds of control relations. For instance, a moral person, who provides a resource to other persons, controls the functionality of this resource. The persons who use this resource have some control on it as well. Each of these actors controls the resource in a different way;
- *depends*: an Actor may depend on another Actor to perform an activity (*e.g.*, a person depends on Google when she accesses her data instances by using the GoogleDocs application);
- *supports*: it means that the target node could never exist without the source node. We may say that the latter allows the former to exist (*e.g.*, a running operating system exists only if it is hosted on a given hardware; an application is supported by the operating system that hosts it; the code of an application supports this application);
- *represents*: it is a relation that exists between data in the social world and their instances on the digital world (*e.g.*, the source code of the Windows operating system is a representation in the digital world of the data known as "Microsoft Windows ©" in the social world).

By applying SOCIOPATH, it is possible to make non-trivial deductions about relations among nodes. For instance, an actor may be able to access digital resources supported by different physical resources connected to each other (*e.g.*, a user can access processes running on different hosts).

Every person owns data in the social world. These data have a concrete existence in the digital world if they are represented by data instances and supported by physical resources. As an actor in the digital world, a person can access and control data instances representing her (and others') data. This may be possibly done through chains of delegations, or by accessing different resources, thus implying some dependence on other persons. In this work, we are particularly interested in formalizing the relations in the digital world, in order to derive the dependences among persons in the social world.

At the present stage of our research, we are focusing on what the users may be able to do, rather than on what they are permitted to do. Thus, imposed access and/or control restrictions are not considered here.

Basic type of instance	The set of all instances		A subset of instances		One instance	
	Notation	*Remark*	*Notation*	*Remark*	*Notation*	*Remark*
Person	$\mathbb{P}$	$\{P : person(P)\}$	$\mathcal{P}$	$\mathcal{P} \subset \mathbb{P}$	P	$P \in \mathbb{P}$
Actors	$\mathbb{A}$	$\{A : actor(A)\}$	$\mathcal{A}$	$\mathcal{A} \subset \mathbb{A}$	A	$A \in \mathbb{A}$
Artifact	$\mathbb{F}$	$\{F : artifact(F)\}$	$\mathcal{F}$	$\mathcal{F} \subset \mathbb{F}$	F	$F \in \mathbb{F}$
Digital resource	$\mathbb{R}$	$\{R : resource(R)\}$	$\mathcal{R}$	$\mathcal{R} \subset \mathbb{R}$	R	$R \in \mathbb{R}$
Physical resource	$\overline{\mathbb{R}}$	$\{\overline{R} : phyresource(\overline{R})\}$	$\overline{\mathcal{R}}$	$\overline{\mathcal{R}} \subset \overline{\mathbb{R}}$	$\overline{R}$	$\overline{R} \in \overline{\mathbb{R}}$
Operating System	$\mathbb{O}$	$\{OS : operatingSystem(OS)\}$	$\mathcal{O}$	$\mathcal{O} \subset \mathbb{O}$	OS	$OS \in \mathbb{O}$
Path	Γ	$\{\sigma : path(\sigma)\}$	Υ	$\Upsilon \subset \Gamma$	σ	$\sigma \in \Gamma$
Activity	$\mathbb{W}$	—	—	—	ω	$\omega \in \mathbb{W}$

Table 1: Glossary of notations

2.4 SOCIOPATH deduction rules and definitions

We use a language based on First Order Logic (FOL) to describe the model of a specific architecture. The edges between nodes are described by binary predicates. Moreover, we propose some rules, based on this language, that formalize the relations in the architecture. Table 1 summarizes all the notations used in the following.

In the remainder of this section, we define and exemplify some deduction rules of SOCIOPATH concerning the relations access and control. These rules are not exhaustive and by no mean we pretend them to capture the whole complexity of a system. They capture several aspects of a simplified vision of the systems that serves the purpose of building an understandable and expressive model.

- An artifact accesses a digital resource, if the artifact can operate the digital resource and the artifact and the digital resource are supported by the same physical resource, or supported by different physical resources connected to each other.

$$\forall F \in \mathbb{F}, \forall R \in \mathbb{R}, \forall \overline{R}1, \overline{R}2 \in \overline{\mathbb{R}} : \bigwedge \left\{ \begin{array}{l} canOperate(F, R) \\ supports(\overline{R}1, F) \\ \bigvee \left\{ \begin{array}{ll} supports(\overline{R}1, R) & \\ \bigwedge \left\{ \begin{array}{l} supports(\overline{R}2, R) \\ isConnectedTo(\overline{R}1, \overline{R}2) \end{array} \right. & \end{array} \right. \end{array} \right. \Rightarrow accesses(F, R) \quad (1)$$

- A person, who owns a physical resource that supports an operating system, accesses this operating system.

$$\forall P \in \mathbb{P}, \forall \overline{R} \in \overline{\mathbb{R}}, \forall OS \in \mathbb{O} : owns(P, \overline{R}) \wedge supports(\overline{R}, OS) \Rightarrow accesses(P, OS) \quad (2)$$

- If an operating system supports and can operate an artifact, it controls this artifact.

$$\forall F \in \mathbb{F}, \forall OS \in \mathbb{O} : supports(OS, F) \wedge canOperate(OS, F) \Rightarrow controls(OS, F) \quad (3)$$

Also we define some concepts we apply during the analysis of a system model. By means of these concepts we can enhance the understanding of the system and ultimately enlighten the actor's dependences on resources and persons (thus addressing both the social world and the digital world). We present some of these definitions.

Definition 1 (Access path) *A path $\sigma$is a list of actors and digital resources such that:*

- $actor(\sigma[1])$*;*
- $\forall i \in [2 : |\sigma|], artifact(\sigma[i]) \wedge accesses(\sigma[i-1], \sigma[i])$*;*
- $resource(\sigma[|\sigma|])$*;*

where $\sigma[i]$, respectively $|\sigma|$, denotes the i^{th} element of σ, respectively the length of σ.
The access paths in the architecture α is noted Υ_α or, simply Υ where there is no ambiguity for α.

Definition 2 (Activities) *Intuitively, an activity is related to an actor who wants to do something concerning a resource (*e.g.*, to access a directory, to copy a file, to edit a document,* etc.*).*

Definition 3 (Actor's dependence on a set of artifacts for an activity) *Let ω be an activity related to an actor A and concerning a resource R. Let $\mathcal{F}$ be a set of artifacts.*

$$A \text{ depends } on \ \mathcal{F} \text{ iff}^1, \forall \sigma \in \Upsilon, \exists F \in \mathcal{F} : F \in \sigma$$

[1] *iff* denotes "if and only if".

Definition 4 (A set of persons controls a set of resources) *Let $\mathcal{R}$ be a set of resources, and $\mathcal{P}$ be a set of persons,*

$$\mathcal{P} \textit{ controls } \mathcal{R} \text{ iff } \wedge \begin{cases} \forall R \in \mathcal{R}, \exists P \in \mathbb{P} : controls(P, R) \Rightarrow P \in \mathcal{P} \\ \forall P \in \mathcal{P}, \exists R \in \mathcal{R} : controls(P, R). \end{cases}$$

Definition 5 (Actor's dependence on a set of persons for an activity) *Let ω be an activity related to an actor A and concerning a resource R. Let $\mathcal{P}$ be a set of persons.*

$$A \textit{ depends on } \mathcal{P} \textit{ for } \omega \text{ iff } \wedge \begin{cases} \exists \mathcal{F} \subset \mathbb{F} : A \textit{ depends on } \mathcal{F} \textit{ for } \omega \\ \mathcal{P} \textit{ controls } \mathcal{F}. \end{cases}$$

3 Use case example: SVN

In the following use case example, we apply SOCIOPATH to a system SVN and deduce some dependences implied by its architecture. What is interesting for us is to show an intelligible way to apply our meta model to a real scenario, rather than presenting surprising outcomes. Figure 2 shows the architecture of a system where the user Philippe uses the application SVN to reach his own document `toto.doc`. The latter is supported by a PC owned by an administrator, and shared with the user Patricia.

In the given scenario, Philippe owns a PC that supports MacOS, which supports an SVN client. In order to keep the figure as readable as possible, a subset of actors involved in the activities are represented (*e.g.*, the persons who provide the aforementioned software are not shown). Philippe's PC can be connected to different networks: Orange (`OrangeNW`) and SFR (`SFRNW`), which are connected to a specific artifact representing the global Internet network (`InternetNW`). Then, the set of local networks also includes the national french university's network (`RenaterNW`) and the one of the University of Nantes specifically (`UnivNantesNW`). The administrator's PC is connected to `UnivNantesNW`. All the providers of these networks appear in our architecture in the social world (except the ones of University of Nantes, for sake of simplicity). Some nodes do not require a fine-grained detailed description (*e.g.* the networks) and are encapsulated in "black boxes", controlled by their providers.
In the following, whenever necessary to avoid ambiguity, we write the name of the resource owner right before the resource name itself (*e.g.*, the SVN Client owned by Philippe may be noted as `PhilippeSVNclient`).

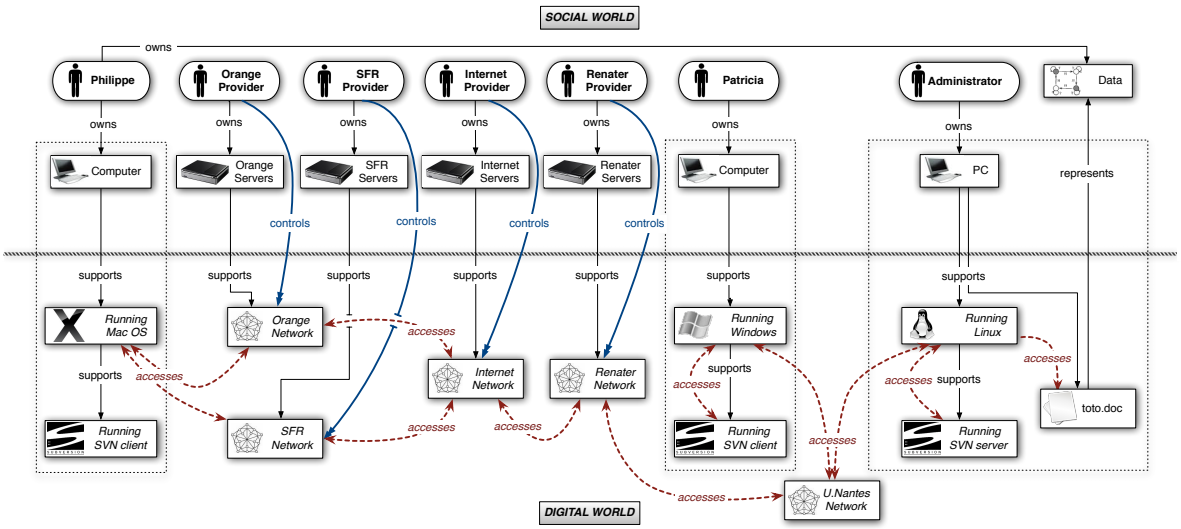

Figure 2: SVN

If we consider the activity "Philippe shares `toto.doc` with Patricia", and by applying the rules on this architecture and analyzing the paths by means of which Philippe and Patricia can access `toto.doc`, we find that Philippe depends on the following sets of persons for the activity "Philippe shares `toto.doc` with Patricia":

`[SFRNW Provider, OrangeNW Provider]`, `[Philippe]`, `[Patricia]`, `[Apple]`, `[SVN Provider]`, `[InternetNW Provider]`, `[RenaterNW Provider]`, `[UnivNantesNW Provider]`, `[Linux Provider]`, `[Administrator]`.

If Patricia does not want to (or is not able to) disclose the details of her architecture to Philippe, they will be encapsulated in a black box, controlled by her.

By analyzing the paths that other persons could have to access `toto.doc`, we conclude that each of the following persons has a possible path to access `toto.doc`:
`[SFRNW Provider]`, `[OrangeNW Provider]`, `[Philippe]`, `[Patricia]`, `[SVN Provider]`, `[InternetNW Provider]`, `[RenaterNW Provider]`, `[UnivNantesNW Provider]`, `[Linux Provider]`, `[Administrator]`.
This makes evident that the administrator can autonomously access `toto.doc`, whereas all the other persons need to collude at least with the administrator, who controls the access to `toto.doc`.

4 Ongoing works and conclusion

This paper introduces SOCIOPATH, a meta-model that formalizes systems in order to reveal the relations of dependence among participants. A formalism of SOCIOPATH is given by several definitions, upon which we have defined some rules, based on first order logic. These rules only captures those aspects of the model that are needed to build the relations among the system's components.

Rules and definitions have been implemented in ProLog, in order to develop a tool, based on SOCIOPATH that infers dependences automatically. Such a tool may be very valuable in all the situations that require a person to evaluate the degree of inter-dependence of the various components of a given architecture.

SOCIOPATH can be used to point out accesses, controls and relations within an architecture. This is particularly useful to check whether the system respects the trust and the privacy expected by its users. Being able to test an architecture compliance with respect to users' privacy policies and trust models is one of our future goals.

We are currently investigating the implication posed by different kinds of control the users may have on system components, with the goal of including access control and restrictions, typical of most common scenarios. A further line of research is devoted to investigate the amount of information about the system, that is needed to derive hidden relations by applying SOCIOPATH. The service level agreements of the system's components, rather than inner design and implementation details (that may not be available or disclosed), should be enough to draw meaningful conclusions.

We believe that the use of our meta-model is not limited to the few possible ways presented above. SOCIOPATH may be used by a standard user, to better analyze and develop useful insights about the digital world, upon which everyone relies more and more.

References

[1] Cornelli, F., Damiani, E., di Vimercati, S.D.C., Paraboschi, S., Samarati, P.: Choosing Reputable Servents in a P2P Network. In: Int. Conference on World Wide Web (WWW). (2002) 376–386

[2] Damiani, E., di Vimercati, D.C., Paraboschi, S., Samarati, P., Violante, F.: A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In: Int. Conference on Computer and Communications Security (CCS). (2002) 207–216

[3] Fahrenholtz, D., Lamersdorf, W.: Transactional Security for a Distributed Reputation Management System. In: Int. Conference on E-Commerce and Web Technologies (EC-WEB). (2002) 214–223

[4] Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The Eigentrust Algorithm for Reputation Management in P2P Networks. In: Int. Conference World Wide Web Conference (WWW). (2003) 640–651

[5] Gupta, M., Judge, P., Ammar, M.: A Reputation System for Peer-to-Peer Networks. In: Int. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV). (2003) 144–152

[6] Carchiolo, V., Longheu, A., Malgeri, M.: Reliable Peers and Useful Resources: Searching for the Best Personalised Learning Path in a Trust and Recommendation-Aware Environment. Information Sciences **180** (2010) 1893–1907

[7] Wang, L., Hill, R.: Trust Model for Open Resource Control Architecture. In: Int. Conference on Computer and Information Technology (CIT). (2010) 817–823

[8] Yoon, J.P., Chen, Z.: Service Trustiness and Resource Legitimacy in Cloud Computing. In: Int. Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC). (2010) 250–257

[9] Varalakshmi, P., Nandini, M., Krithika, K., Aarthi, R.: An Optimal Trust Based Resource Allocation Mechanism for Cross Domain Grid. In: Int. Conference on Recent Trends in Business Administration and Information Processing (BAIP). (2010) 342–348

[10] Alhadad, N., Lamarre, P., Busnel, Y., Serrano-Alvarado, P., Biazzini, M., Sibertin-Blanc, C.: SocioPath: In Whom You Trust? Technical Report hal-00608435, LINA (Laboratoire d'Informatique de Nantes Atlantique). (2011)